Thoughts on Curriculum for Embedded Computing

Wayne Wolf
Dept. of Electrical Engr.
Princeton University

Outline

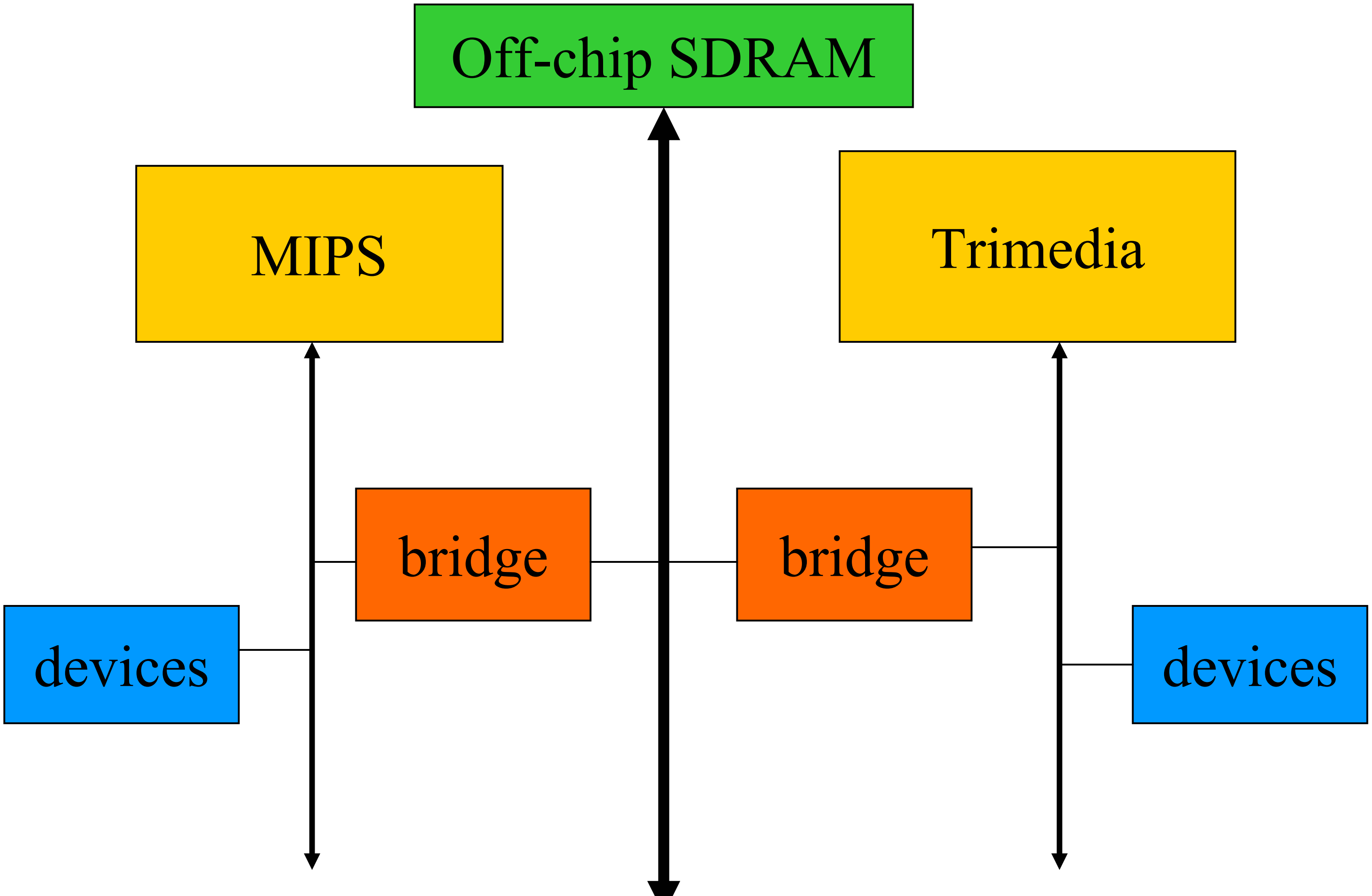
- Characteristics of embedded systems.
- Graduate education.
- Undergraduate education.

Characteristics of embedded systems

Examples

- Radar processing system.
- Automotive engine control.
- PDA.
- Set-top box.
- Smart camera.

Viper set-top-box chip



Typical requirements

- Complex functionality.
- Real-time.
- Multi-rate.
- Often low power.
- Low manufacturing cost.

Embedded system design

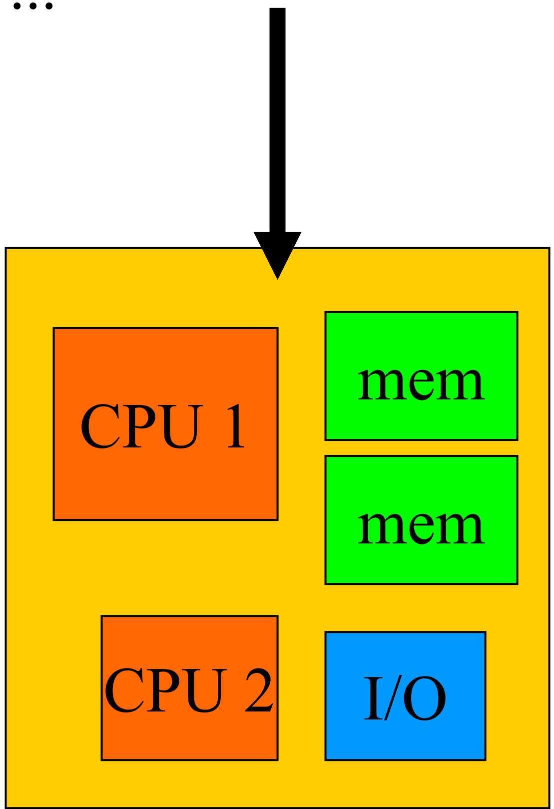
- Software doesn't do anything without hardware.
  - Hardware is the lens through which we view software characteristics.
- Software executes on a hardware platform.

The platform view

application software

```
for (I=0; I<N; I++)
   a[I] = b[I]*c[I];
...
```

hardware platform



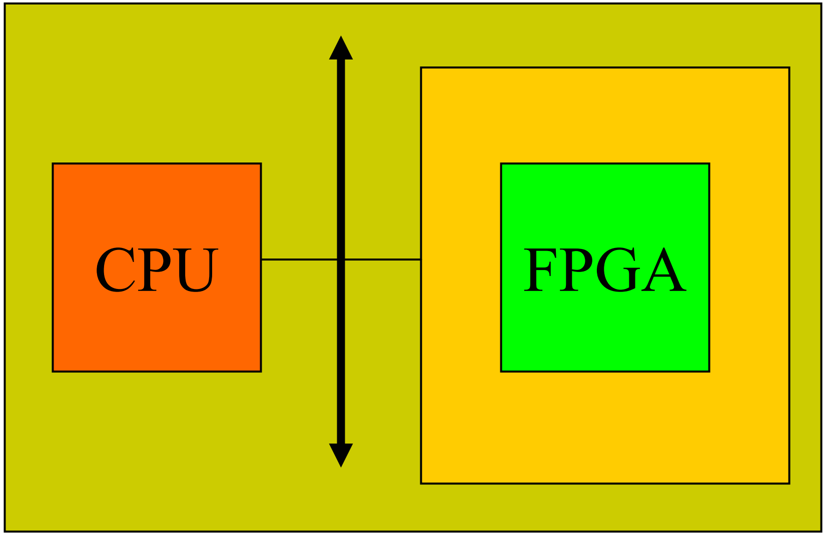
Graduate education

Applications are critical

- Embedded computers are used in many ways.
  - Applications vary considerably in their characteristics.
- Application characteristics affect:
  - Models of computation.
  - Constraints.
  - Software and hardware architecture.
  - Detailed optimizations.
  - Design methodologies.
- All students should know something about at least one application area.

HW/SW partitioning classes

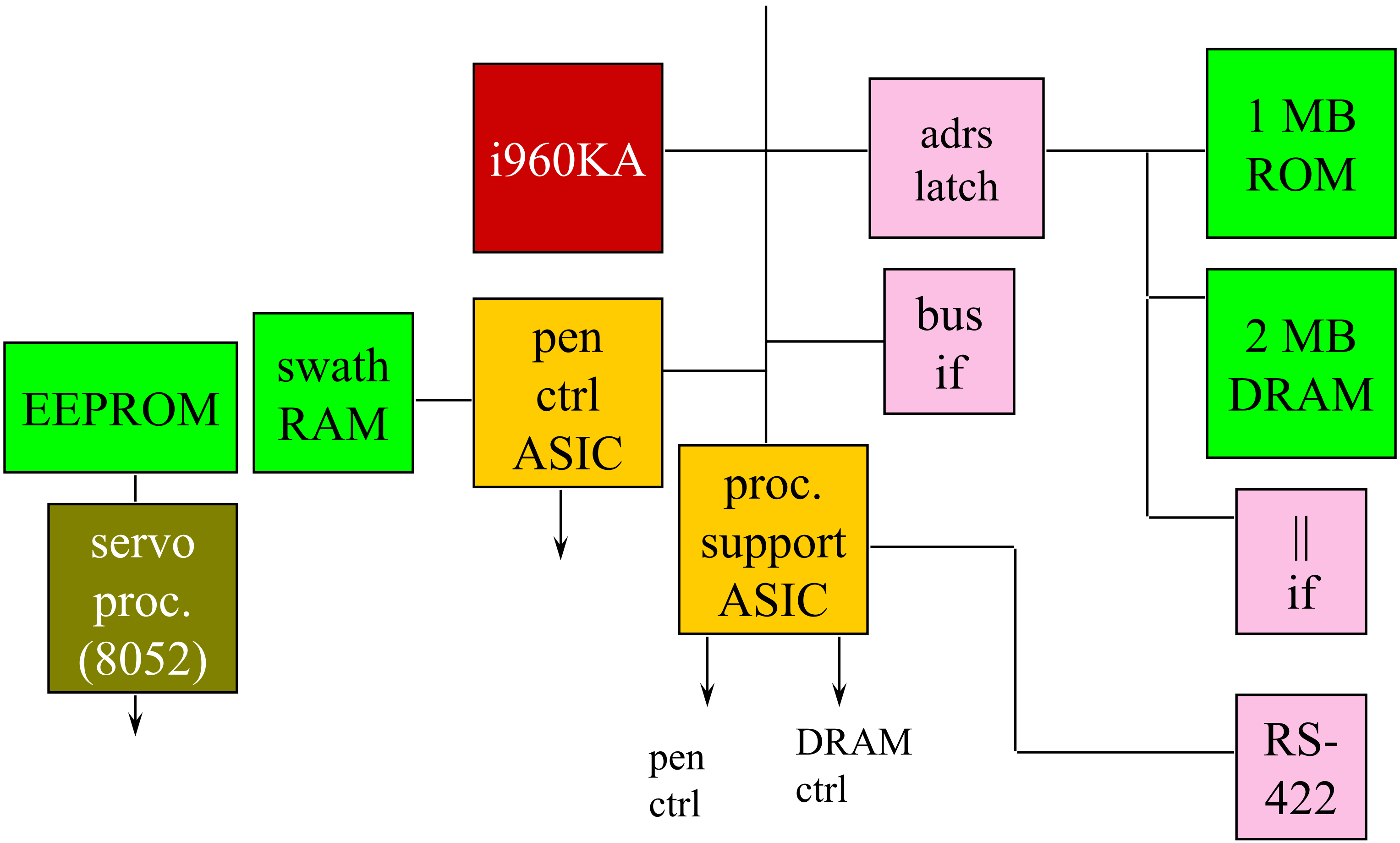
- Many VLSI classes have morphed into co-design courses using FPGA plug-in boards:



Why distributed systems?

- CPU cost is a non-linear function of performance.
  - Several small CPUs may be cheaper than one big one.
- Scheduling overhead must be paid for at the non-linear rate.

HP DesignJet architecture



Standards-based embedded systems

- Many product categories rely on standards.
- Standards body provides reference implementation.
  - Reduces development time.
  - Don't want to introduce bugs.
- Reference implementation may not be well-suited to implementation:
  - No task structure;
  - Not optimized.

Separation of concerns

- Traditional software design emphasizes function.
- Embedded computing allows us to move real time/low power design from hardware to software.
- Requires us to strengthen software design methodologies.

Design challenges

- Must design new systems quickly.
- Must incorporate existing complex components.
- Must produce highly optimized implementations.
- Must produce a believably reliable design.

Areas of study

- Hardware.
  - Multiprocessors.
  - Performance.
  - Power/energy.
- Software.
  - Models of computation.
  - Programs.
  - Real-time.
- Systems.
  - The computer/real-world interface.
  - Design methodologies and standards.
  - Algorithms and mapping.

Undergraduate education

What every EE/CS student should know

- Embedded computers are everywhere.
- Computing is a physical process:
  - Time.
  - Energy.
- Computers provide real and apparent concurrency.
- Embedded programs must work through the mechanisms of computers to provide timely, energy-efficient computing.

Microprocessor course considered harmful

- Traditional microprocessor-based courses emphasized hardware and details.
- Embedded systems courses teach software+hardware, general principles.
- Grounding in particulars is still good for students, but with an eye toward generalization.

CPU characteristics and systems

- Engine design dictates platform design.
- Important characteristics:
  - performance;
  - power;
  - code size
    - main memory footprint;
    - cache footprint.

Basics to be taught in embedded course

- Fundamentals of the course's example processor.
- Relationship of architectural characteristics to:
  - performance;
  - power;
  - code size.

Performance measurement

- Architecture view is machine-centric:
  - how do instructions affect the pipeline? the cache?
- Systems view is program-centric:
  - how do the machine characteristics affect this part of the program?
  - How does this relate to the overall program?

The performance analysis equation

- Original work done by Shaw:
  - performance = path analysis + path timing
- Path analysis requires analysis of program.
- Path timing requires instruction-level model.

Single-program performance optimization

- Pipeline timing can have some effect.
- Caches have major effects:
  - code;
  - data.
- Is it as fast as we need it to be?
  - Where is the bottleneck?
  - How much can bottlenecks be speeded up?

Power analysis

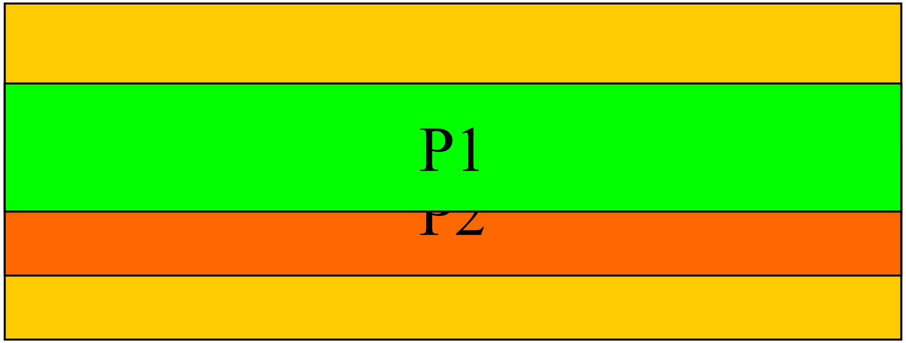
- To a first approximation, high speed = low power.
- Good news, since instruction-level power models are hard to come by.
- Proper analysis includes:
  - CPU core;
  - cache;
  - busses.

Multiple processes

- Many designs use multiple processes:
  - multi-rate deadlines;
  - different types of behavior.
- Multiple processes complicate all phases of analysis.
  - Instruction-level models are often too cumbersome.

Multiple processes on one CPU

- RTOS schedules multiple processes on CPU.
- Non-ideal effects:
  - context switching;
  - cache interference.



Methodologies

- Requirements:
  - imprecise
- Specifications:
  - precise
- Quality assurance
  - a management problem
  - requires verification at all levels of abstraction

Real-world examples

- Hard to get but very valuable.
- Lessons:
  - complexity;
  - design methodology;
  - architectural strategies;
  - verification methods.

Summary

- Every EE/CS undergraduate should know a little about embedded computing.
- Every computer systems undergraduate should know quite a bit about embedded computing.
- We need more graduate courses in embedded computing:
  - New material.
  - Tailored existing material.